Fig. 1

Fig. 2

Fig. 3

1. Accessing DataCollection: Part 42

(cf. Fig. 1)

Markus Demleitner
msdemlei@ari.uni-heidelberg.de

(cf. Fig. 2)

- The Problem
- Not-Solutions
- Another Attempt

(cf. Fig. 3)

2. The Problem

Dozens of data collections ("individuals") in a single service ("aggregate"), as in

- TAP services
- (specifically ObsTAP)
- Aggregated SIAP/SSAP endpoints

We want *simultaneously*

- Locate individualx by their metadata ("images from MINDSTEp?") and figure out how to query them
- Allow all-VO queries ("are there images at 50,+34?") without hitting aggregates multiple times

3. Non-Solutions I

First attempt: Register individuals as DataCollections, have served-by relationships to their aggregates.

Pro: It's clean.

Against:

- Complicated query patterns if these are to be handled like "normal" services.
- Splits the schema into metadata belonging to the aggregate ("capability-bound") and belonging to the individual ("resource-bound") and then some, making the thing conceptually difficult.

There'd be a fairly easy way out: Just register all resources divided up in DataCollection and DataService, linked through relationships. Realistically, I think it won't happen, as that'd uproot Registry practices of the VO's lifetime, and also almost double the number records currently in the registry.

4. Non-Solutions II

Second attempt: Just add the capability elements of the aggregate to the individual.

Pro: It's simple, query patterns symmetrical in individuals and plain services

Against:

- VODataService schema change required
- existing queries for all-VO searches would need to be changed to restrict results to non-DataCollection records.
- Capability elements (e.g., for TAP) may be much more complex than useful for this use case.

The plan made in Madrid to just try using normal DataServices for that blew up right away when I tried it: Client writers immediately complained, even for TAP where I'd have expected things to hold up for a while. What actually broke things is what retrospectively was a design error: dataModel should have been in the resource, not the capability.

5. Better Luck This Time?

Current plan: Auxiliary capabilities.

Only the aggregate would have a capability with the normal id (e.g., `ivo://ivoa.net/std/sia`)

The idividuals have very basic capabilities with ids like `ivo://ivoa.net/std/sia#aux`.

Pro: Backwards-compatible. Simple query patterns (use LIKE). Schema change not absolutely required.

Against: Ugly. Maybe not as the night, but ugly. Also: Where do we specify this? The additional ids would have to be in the actual standards and their records.